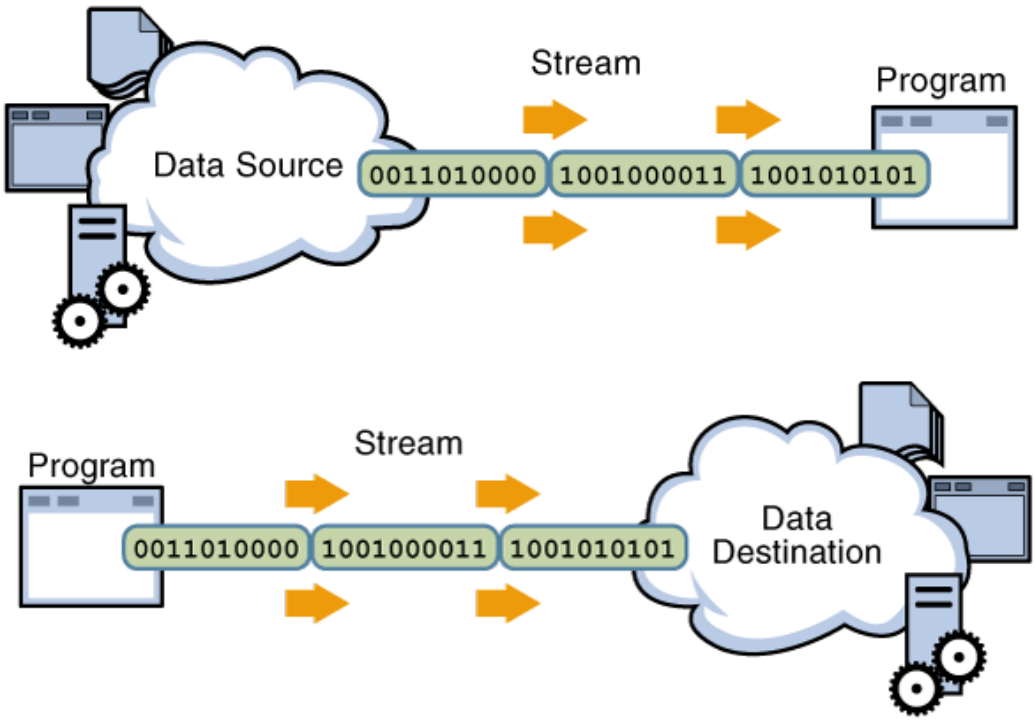
Ejercicios del ANEXO



1. Ejercicios

1. (`CuentaLineas`) Escribe un programa que, sin utilizar la clase `Scanner`, muestre el número de lineas que contiene un fichero de texto. El nombre del fichero se solicitará al usuario al comienzo de la ejecución.
2. (`CuentaPalabras`) Escribe un programa que, sin utilizar la clase `Scanner`, muestre el número de palabras que contiene un fichero de texto. El nombre del fichero se solicitará al usuario al comienzo de la ejecución.

   > Sugerencia: Lee el fichero, línea a línea y utiliza la clase `StringTokenizer` o bien el método `split` de la clase `String` para averiguar el nº de palabras.

3. (`Censura`) Escribir un programa que sustituya por otras, ciertas palabras de un fichero de texto. Para ello, se desarrollará y llamará al método `void aplicaCensura(String entrada, String censura, String salida),` que lee de un fichero de entrada y mediante un fichero de censura, crea el correspondiente fichero modificado. Por ejemplo:

   Fichero de entrada:

   ```
   En un lugar de la Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo de los de lanza en astillero
   ```

   Fichero de censura:

   ```
   lugar sitio
   quiero debo
   hidalgo noble
   ```

   Fichero de salida:

   ```
   En un sitio de la Mancha, de cuyo nombre no debo acordarme, no ha mucho tiempo que vivía un noble de los de lanza en astillero
   ```

   > Valora la posibilidad de cargar el fichero de censura en un mapa o par clave, valor.

4. (`Concatenar1`) Escribe un programa que dados dos ficheros de texto `f1` y `f2` confeccione un tercer fichero `f3` cuyo contenido sea el de `f1` y a continuación el de `f2`.
5. (`Concatenar2`) Escribe un programa que dados dos ficheros de texto `f1` y `f2`, añada al final de `f1` el contenido de `f2`. Es decir, como el ejercicio anterior, pero sin producir un nuevo fichero.
6. (`Iguales`) Escribir un programa que compruebe si el contenido de dos ficheros es idéntico. Puesto que no sabemos de qué tipo de ficheros se trata, (de texto, binarios, ...) habrá que hacer una comparación byte por byte.
7. Escribe los siguientes métodos y programa:
   - Método `void generar()` que genere 20 números aleatorios enteros entre 1 y 100 y los muestre por pantalla.

- Método `void media()` que lea de teclado 20 números enteros y calcule su media.
- Programa que, modificando la entrada y la salida estándar, llame a `generar()` para que los datos se graben en un fichero y a continuación llame a `media()` de manera que se tomen los datos del fichero generado.

8. (`Notas`) Escribir un programa que almacene en un fichero binario (`notas.dat`) las notas de 20 alumnos. El programa tendrá el siguiente funcionamiento:
    - En el fichero se guardarán como máximo 20 notas, pero se pueden guardar menos. El proceso de introducción de notas (y en consecuencia, el programa) finalizará cuando el usuario introduzca una nota no válida (menor que cero o mayor que 10).
    - Si, al comenzar la ejecución, el fichero ya contiene notas, se indicará al usuario cuántas faltan por añadir y las notas que introduzca el usuario se añadirán a continuación de las que hay.
    - Si, al comenzar la ejecución, el fichero ya contiene 20 notas, se le preguntará al usuario si desea sobrescribirlas. En caso afirmativo las notas que introduzca sustituirán a las que hay y en caso negativo el fichero no se modificará.

2. Los flujos estándar

1. (clase `LeeNombre`) Escribir un programa que solicite al usuario su nombre y, utilizando directamente `System.in`, lo lea de teclado y muestre por pantalla un mensaje del estilo "Su nombre es Miguel". Recuerda que `System.in` es un objeto de tipo `InputStream`. La clase `InputStream` permite **leer bytes** utilizando el método `read()`. Será tarea nuestra ir construyendo un `String` a partir de los bytes leídos. Prueba el programa de manera que el usuario incluya en su nombre algún carácter “extraño”, por ejemplo el símbolo "€" ¿Funciona bien el programa? ¿Por qué?
2. (clase `LeeEdad`) Escribir un programa que solicite al usuario su edad y, utilizando directamente `System.in`, la lea de teclado y muestre por pantalla un mensaje del estilo "Su edad es 32 años". En este caso, será tarea nuestra construir un String a partir de los bytes leídos y transformarlo posteriormente en un entero.
3. (clase `CambiarEstandar`). La salida estándar (`System.out`) y la salida de errores (`System.err`) están asociadas por defecto con la pantalla. Se puede cambiar este comportamiento por defecto utilizando los métodos `System.setout` y `System.seterr` respectivamente. Investiga un poco cómo se utilizan, escribe un programa que asocie la salida estándar a al fichero `salida.txt` y la salida de errores al fichero `errores.txt` y, a continuación, escriba algún mensaje en cada uno de las salidas, por ejemplo `System.out.println(“El resultado es 20”);` y `System.err.println(“ERROR: Elemento no encontrado”);`
4. (`SumarEdades`)
    - Escribir método `void sumaEdades()` que lea de teclado las edades de una serie de personas y muestre cuanto suman. El método finalizará cuando el usuario introduzca una edad negativa.
    - Escribir un método `main` que llame al método anterior para probarlo.
    - Modificar el método `main` de forma que, antes de llamar al método `sumaEdades`, se cambie la entrada estándar para que tome los datos del fichero `edades.txt` en lugar de leerlos de teclado.

3. `InputStreamReader`

5. (leerByte) `System.in` (`InputStream`) está orientado a lectura de bytes. Escribe un programa que lea un byte de teclado y muestre su valor (int) por pantalla. Pruébalo con un carácter "extraño", por ejemplo '€'.
6. (leerCaracter) `InputStreamReader` (`StreamReader`) está orientado a carácteres. Escribe un programa que lea un carácter de teclado usando un `InputStreamReader` y muestre su valor (`int`) por pantalla. Pruébalo con un carácter "extraño", por ejemplo '€'. ¿Se obtiene el mismo resultado que en el ejercicio anterior?.

4. Entrada "orientada a líneas".

En los ejercicios anteriores, las limitaciones de la clase utilizada (`InputStream`), nos obliga a incluir en el programa instrucciones que detecten que el usuario ha terminado su entrada (ha pulsado **INTRO**). La clase `BuffereReader` dispone del método `readLine()`, capaz de leer una línea completa (la propia instrucción detecta el final de la línea) y devolver un `String`.

7.- Repite el ejercicio 1 utilizando un `BufferedReader` asociado a la entrada estándar. La clase `BufferedReader`, está orientada a leer caracteres en lugar de bytes. ¿Qué ocurre ahora si el usuario introduce un carácter "extraño" en su nombre?

8.- Repite el ejercicio 2 utilizando un `BufferedReader` asociado a la entrada estándar.

5. Lectura/escritura en ficheros

9. (EscribirFichero1) Escribe un programa que, usando las clases `FileOutputStream` y `FileInputStream`,
    - escriba los caracteres de tu nombre en un fichero (`nombre.txt`).
    - lea el fichero creado y lo muestre por pantalla.
    - Si abrimos el fichero creado con un editor de textos, ¿su contenido es legible?
10. (EscribirFichero2) Repetir el ejercicio anterior utilizando las clases `FileReader` y `FileWriter`.

6. Uso de buffers

Los buffers hacen que las operaciones de lectura-escritura se realicen inicialmente en memoria y, cuando los buffers correspondientes están vacíos/llenos, se hagan definitivamente sobre el dispositivo.

11. (TestVelocidadBuffer) Vamos a probar la diferencia de tiempo que conlleva escribir datos a un fichero directamente o hacerlo a través de un buffer. Para ello, crea un fichero de 1 Mb (1000000 de bytes aprox.) usando un la clase `FileWriter` y mide el tiempo que tarda en crearlo. Posteriormente, crea un fichero de exactamente el mismo tamaño utilizando un `BufferedWriter` y mide el tiempo que tarda. ¿Hay diferencia?. Para medir el tiempo puedes utilizar `System.currentTimeMillis()`, inmediatamente antes y después de crear el fichero y restar los valores obtenidos.
12. (TestVelocidadBuffer2) Modifica el programa `TestVelocidadBuffer` para probar cómo afecta a la escritura con buffer la ejecución de la instrucción `flush()`. Esta instrucción fuerza el volcado del buffer a disco. ¿Disminuye la velocidad si tras cada operación de escritura ejecutamos flush()?
13. (TestVelocidadBuffer3) Modifica el programa `TestVelocidadBuffer` para probar cómo a la velocidad el tamaño del buffer. La clase `BufferedWriter` tiene un constructor que permite indicar el tamaño del buffer. Prueba con distintos valores.

7. Streams para información binaria

14. (Personas) Escribe un programa que, utilizando entre otras la clase `DataOutputStream`, almacene en un fichero llamado `personas.dat` la información relativa a una serie de personas que va introduciendo el usuario desde teclado:
    - `Nombre` (String)
    - `Edad` (entero)
    - `Peso` (double)
    - `Estatura` (double)

    La entrada del usuario terminará cuando se introduzca un nombre vacío.

    Nota: Utiliza la clase `Scanner` para leer desde teclado y los métodos `writeDouble`, `writeInt` y `writeUTF` de la clase `DataOutput/InputStream` para escribir en el fichero)

    Al finalizar el programa, abre el fichero resultante con un editor de texto (notepad o wordpad) ¿La información que contiene es legible?.
15. (AñadirPersonas) Modifica el programa anterior para que el usuario, al comienzo del programa, pueda elegir si quiere añadir datos al fichero o sobre escribir la información que contiene.
16. (MostrarPersonas) Realizar un programa que lea la información del fichero `personas.dat` y la muestre por pantalla. Para determinar que no quedan más datos en el fichero podemos capturar la excepción `EOFException`
17. (CalculosPersonas) Realizar un programa, similar al anterior, que lea la información del fichero `personas.dat` y muestre por pantalla la estatura que tienen de media las personas cuya edad está entre 20 y 30 años.

8. Streams de objetos. Serialización.

18. (GuardaLibros)
    - (Autor) Crea la clase autor, con los atributos nombre, año de nacimiento y nacionalidad. Incorpora un constructor que reciba todos los datos y el método `toString()`.
    - (Libro) Crea la clase Libro, con los atributos titulo, año de edición y autor (Objeto de la clase autor). Incorpora un constructor que reciba todos los datos y el método `toString()`.
    - Escribe un programa (GuardaLibros) que cree tres libros y los almacene en el fichero `biblioteca.obj`.
    - Las clases deberán implementar el interfaz `Serializable`.
19. (LeeLibros) Escribe un programa que lea los objetos del fichero `biblioteca.obj` y los muestre por pantalla.

9. Más ejercicios (Lionel)

> Para probar algunos de estos ejercicios debes utilizar el archivo `Documentos.zip`. Descárgalo del aula virtual y descomprímelo en la carpeta de cada proyecto que crees.

21. **Mostrar información de ficheros**

Implementa un programa que pida al usuario introducir por teclado una ruta del sistema de archivos (por ejemplo, `C:/Windows` o `Documentos`) y muestre información sobre dicha ruta (ver función más abajo). El proceso se repetirá una y otra vez hasta que el usuario introduzca una ruta vacía (tecla intro). Deberá manejar las posibles excepciones.

Necesitarás crear la función `void muestraInfoRuta(File ruta)` que dada una ruta de tipo `File` haga lo siguiente:

- Si es un archivo, mostrará por pantalla el nombre del archivo.
- Si es un directorio, mostrará por pantalla la lista de directorios y archivos que contiene (sus nombres). Deberá mostrar primero los directorios y luego los archivos.
- En cualquier caso, añade delante del nombre la etiqueta `[*]` o `[A]` para indicar si es un directorio o un archivo respectivamente.
- Si el path no existe lanzará un `FileNotFoundException`.

22. **Mostrar información de ficheros (v2)**

Partiendo de una copia del programa anterior, modifica la función `muestraInfoRuta`:

- En el caso de un directorio, mostrará la lista de directorios y archivos en orden alfabético. Es decir, primero los directorios en orden alfabético y luego los archivos en orden alfabético. Te será útil `Arrays.sort()`.
- Añade un segundo argumento `boolean info` que cuando sea `true` mostrará, junto a la información de cada directorio o archivo, su tamaño en bytes y la fecha de la última modificación. Cuando `info` sea `false` mostrará la información como en el ejercicio anterior.

23. **Renombrando directorios y ficheros**

Implementa un programa que haga lo siguiente:

- Cambiar el nombre de la carpeta `Documentos` a `DOCS`, el de la carpeta `Fotografias` a `FOTOS` y el de la carpeta `Libros` a `LECTURAS`
- Cambiar el nombre de todos los archivos de las carpetas `FOTOS` y `LECTURAS` quitándole la extensión. Por ejemplo, `astronauta.jpg` pasará a llamarse `astronauta`.

24. **Creando (y moviendo) carpetas**

Implementa un programa que cree, dentro de `Documentos`, dos nuevas carpetas: `Mis Cosas` y `Alfabeto`. Mueve las carpetas `Fotografias` y `Libros` dentro de `Mis Cosas`. Luego crea dentro de `Alfabeto` una carpeta por cada letra del alfabeto (en mayúsculas): `A`, `B`, `C`... `Z`. Te serán de ayuda los códigos numéricos ASCII: https://elcodigoascii.com.ar

25. **Borrando archivos**

Implementa un programa con una función `boolean borraTodo(File f)` que borre `f`: Si no existe lanzará una excepción. Si es un archivo lo borrará. Si es un directorio, borrará primero sus archivos y luego el propio directorio (recuerda que para poder borrar un directorio debe estar vacío). Devolverá `true` si pudo borrar el `File f` (`false` si no fué posible).

Prueba la función borrando las carpetas: `Documentos/Fotografias`, `Documentos/Libros` y `Documentos` (es decir, tres llamadas a la función, en ese orden).

Super extra challenge: Esta función, tal y como está definida, no borrará las subcarpetas que estén dentro de una carpeta (para ello habría que borrar primero el contenido de dichas subcarpetas). ¿Se te ocurre cómo podría hacerse?

26. **Máximo y mínimo**

Implementa un programa que muestre por pantalla los valores máximos y mínimos del archivo `numeros.txt`.

27. **Notas de alumnos**

El archivo `alumnos_notas.txt` contiene una lista de 10 alumnos y las notas que han obtenido en cada asignatura. El número de asignaturas de cada alumno es variable. Implementa un programa que muestre por pantalla la nota media de cada alumno junto a su nombre y apellido, ordenado por nota media de mayor a menor.

28. **Ordenando archivos**

Implementa un programa que pida al usuario un nombre de archivo `A` para lectura y otro nombre de archivo `B` para escritura. Leerá el contenido del archivo `A` (por ejemplo `usa_personas.txt`) y lo escribirá ordenado alfabéticamente en B (por ejemplo `usa_personas_sorted.txt`).

29. **Nombre y apellidos**

Implementa un programa que genere aleatoriamente nombres de persona (combinando nombres y apellidos de `usa_nombres.txt` y `usa_apellidos.txt`). Se le pedirá al usuario cuántos nombres de persona desea generar y a qué archivo **añadirlos** (por ejemplo `usa_personas.txt`).

30. **Diccionario**

    Implementa un programa que cree la carpeta `Diccionario` con tantos archivos como letras del abecedario (`A.txt`, `B.txt`... `Z.txt`). Introducirá en cada archivo las palabras de `diccionario.txt` que comiencen por dicha letra.

31. **Búsqueda en PI**

    Implementa un programa que pida al usuario un número de cualquier longitud, como por ejemplo "1234", y le diga al usuario si dicho número aparece en el primer millón de decimales del nº pi (están en el archivo `pi-million.txt`). No está permitido utilizar ninguna librería ni clase ni método que realice la búsqueda. Debes implementar el algoritmo de búsqueda tú.

32. **Estadísticas**

    Implementa un programa que lea un documento de texto y muestre por pantalla algunos datos estadísticos: nº de líneas, nº de palabras, nº de caracteres y cuáles son las 10 palabras más comunes (y cuántas veces aparecen). Prueba el programa con los archivos de la carpeta `Libros`.

NOTA: *Para llevar la cuenta de cuántas veces aparece cada palabra puedes utilizar una HashTable. Una tabla hash es una estructura de datos tipo colección (como el ArrayList), que permite almacenar pares clave-valor. Por ejemplo {"elefante", 5} o {"casa", 10} son pares <String,Integer> que asocian una palabra (clave) con un nº entero (valor).*

10. Fuentes de información

- Wikipedia
- Programación (Grado Superior) - Juan Carlos Moreno Pérez (Ed. Ra-ma)
- Apuntes IES Henri Matisse (Javi García Jimenez?)
- Apuntes AulaCampus
- Apuntes José Luis Comesaña
- Apuntes IOC Programació bàsica (Joan Arnedo Moreno)
- Apuntes IOC Programació Orientada a Objectes (Joan Arnedo Moreno)
- Apuntes Lionel